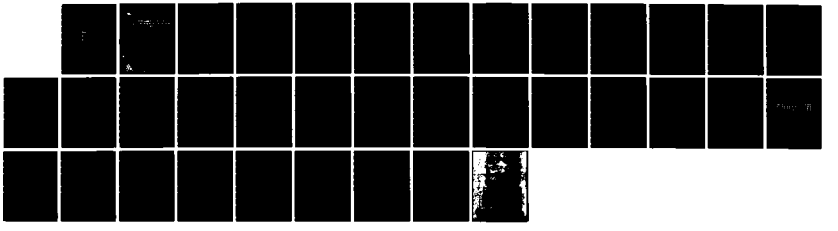
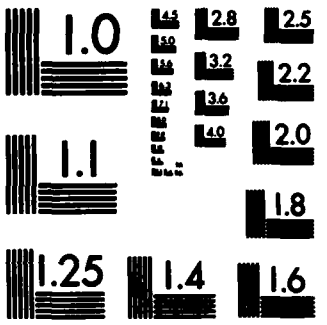
AD-A144 001 PROGRAMMING GUIDE FOR SHIPBOARD NUMERICAL AND PROGRAMS (SNAP)(U) NAVAL ENVIRONMENTAL PREDICTION RESEARCH FACILITY MONTEREY CA T BROWN JUN 84 NEPRF-TR-84-06 1/1

UNCLASSIFIED F/G 9/2 NL





NAVENVPREDRSCHFAC
TECHNICAL REPORT
TR 84-06

(12)

PROGRAMMING GUIDE FOR SHIPBOARD NUMERICAL AID PROGRAMS (SNAP)

Terry Brown
Naval Environmental Prediction Research Facility

JUNE 1984

NAVAL ENVIRONMENTAL PREDICTION RESEARCH FACILITY
MONTEREY, CALIFORNIA 93943

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NAVENVPREDRSCHFAC Technical Report TR 84-06	2. GOVT ACCESSION NO. AD-A144001	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* Programming Guide for Shipboard Numerical Aid Programs (SNAP)		5. TYPE OF REPORT & PERIOD COVERED Final
		6. PERFORMING ORG. REPORT NUMBER TR 84-06
7. AUTHOR(s) Terry Brown		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Environmental Prediction Research Facility Monterey, CA 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE 62759N PN WF59-551 NEPRF WU 6.2-34
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Air Systems Command Department of the Navy Washington, DC 20361		12. REPORT DATE June 1984
		13. NUMBER OF PAGES 32
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Programming guide
BASIC
Human Factors

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

Guidelines for programming meteorological applications software in BASIC on a Hewlett-Packard 9845 desktop computer are given. These guidelines include control structure, programming conventions, and a human factors style guide.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

CONTENTS

RECORD OF CHANGES

CHANGE NUMBER	DATE OF CHANGE	DATE ENTERED	PAGE NUMBER	ENTERED BY

1. SCOPE

1.1 PURPOSE

This document establishes a programming guide for all BASIC code developed in support of the Shipboard Numerical Aid Program (SNAP) project. This document illustrates how the guidance of MIL-STD-1679 shall be applied to SNAP software and is modeled after reference [2], the FORTRAN Code Development Standard for the Tactical Environmental Support System (TESS).

1.2 APPLICATION

This guide shall apply to all BASIC code written for the Hewlett-Packard 9845 desktop computer as part of the SNAP project.

In this document the word "shall" conveys a requirement while the word "should" conveys a suggestion.

2. REFERENCE DOCUMENTS

[1]. MIL-STD-1679, Weapon System Software Development, 1 December 1978.

[2]. FORTRAN Code Development Standard for Tactical Environmental Support System, August 1982.

3. GENERAL REQUIREMENTS

3.1 DESIGN REQUIREMENTS

Each SNAP shall be designed to meet the requirements described in the applicable Program Performance Specification (PPS) if such document exists. The PPS describes in detail all of the operational and functional requirements necessary to design a specific computer program.

3.2 PROGRAM GENERATION

All SNAP software shall be coded in HP-BASIC on a Hewlett-Packard 9845 desktop computer. The use of the word BASIC in this document shall mean HP-BASIC. Code preferably should run on both the A and B models of the 9845 computer, hereafter called the 9845A and 9845B. Writing separate 9845A and 9845B versions of a program shall be avoided.

In some cases, a program may be written specifically for the 9845B with Option 275 installed, hereafter called the 9845B-275. Such programs will not run on a 9845A or a 9845B that does not have Option 275. Programs shall be written specifically for the 9845B-275 only when a tactically useful version of the program cannot be developed for the 9845A or 9845B.

4. DETAILED REQUIREMENTS

4.1 PROGRAM DESIGN REQUIREMENTS

Although BASIC is not a structured programming language, SNAP software shall adhere as much as practicable to the structured programming principles listed in Appendix B (glossary) of this document. Appendix C contains examples of simulated structured programming constructions.

4.2 PROGRAMMING GUIDELINES

The following design and coding guidelines shall apply to the development of SNAP software.

4.2.1 Control Structures

Programs should be designed and coded using the three fundamental control structures presented in figures 1 through 3; sequence of operations (assignment, arithmetic operation, ...); conditional branch (IF...THEN...); and repetition of operations (FOR...NEXT).

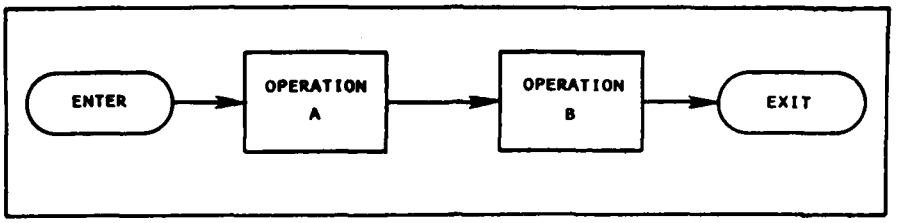


Figure 1. Sequence of operations.

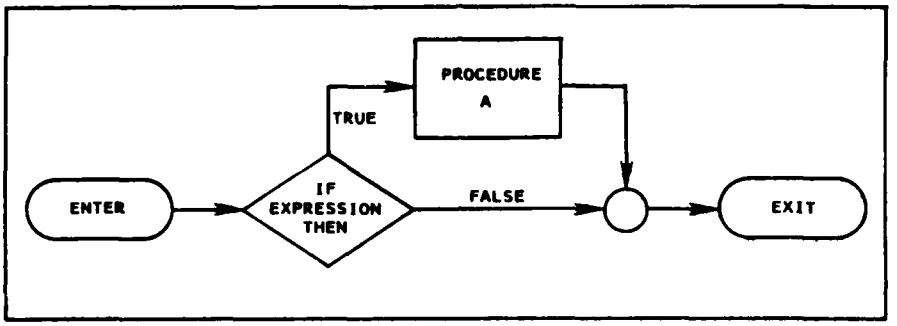


Figure 2. Conditional branch.

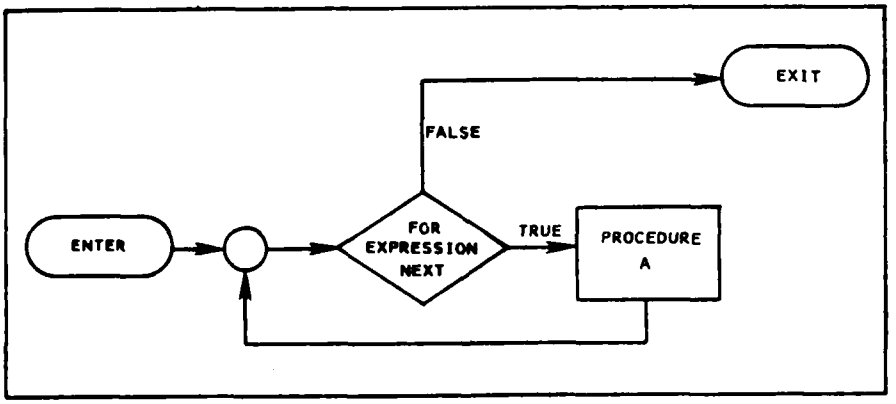


Figure 3. Repetition of operations.

A Structured Programming ROM is included as part of Option 275 for the 9845B. This ROM extends the BASIC language to include statements for structured looping (WHILE...END WHILE; REPEAT...UNTIL) and structured decisions (IF...THEN...ELSE; SELECT...CASE). Since these statements are not available for the 9845A, their use shall be limited to programs designed specifically for the 9845B-275.

4.2.2 Entry-Exit Structure

Each program segment (see glossary for definition) must have one entry and one exit. The one entry shall be at the first executable statement of the segment. The one exit shall be a SUBEND, RETURN, or FNEND statement at the end of the segment.

4.2.3 Size

To avoid SNAP to TESS program conversion problems, the main program and all subprograms should not exceed one hundred executable statements. This conforms to a requirement in reference [2]. In addition, the use of many short program segments, rather than a few large segments, results in better computer memory packing efficiency.

4.2.4 Branching

Use only simple integers as the index of a computed GO TO. For example,

```
ON Index GO TO 1500,2000
```

not

```
ON Index(I) GO TO 1500,2000
```

Note that the computed GO TO is similar to, but is not the same as the CASE control structure.

Do not use a branching statement (GO TO; ON Index GO TO) to pass control to a statement that is in a different program segment.

4.2.5 Indentation

Indentation of program statements shall be used to improve readability. The Structured Programming ROM included in the 9845B-275 contains an indentation utility which performs this formatting. Programmers without access to this utility (9845A users, for example) may submit unindented code which will be indented at NEPRF using the Structured Programming ROM indentation utility. Figure 4 shows an example of the Structured Programming ROM indentation scheme.

Before Indention

```
10    ! This is an example of a program BEFORE indentation
20    !
30 Start: ! of program
40    !
50    REAL R,C,B(3,3)
60    INTEGER I,J
70    !
80    PRINT "No indentation example"
90    !
100   FOR I=1 TO 3
101   PRINT "I=";I
110   FOR J=1 TO 3
111   PRINT "J=";J
120   B(I,J)=0
140   NEXT J
141   PRINT "INITIALIZED ",I,J
150   NEXT I
151   !
160 Input_1:  ! enter data
161   !
170   INPUT "ENTER RADIUS OF CIRCLE",R
180   IF R>0 THEN Compute
190   BEEP
200   PRINT R;" IS NOT A VALID RADIUS ENTRY"
210   GOTO Input_1
220   !
230 Compute:  ! continue
240   !
250   C=FNCirc(R)
260   CALL Print(C,R)
270   PRINT "END"
280   END
281   !
290   DEF FNCirc(R)
300   C=2*R*PI
310   RETURN C
320   FNEND
321   !
330   SUB Print(C,R)
340   PRINT "RADIUS     CIRCUMFERENCE"
350   PRINT R,C
360   SUBEND
380   !
```

After Indention

```
10    ! This is an example of a program AFTER indentation
20    !
30 Start:   ! of program
40    !
50    REAL R,C,B(3,3)
60    INTEGER I,J
70    !
80    PRINT "   Indentation example"
90    !
100   FOR I=1 TO 3
101     PRINT "I=";I
110     FOR J=1 TO 3
111       PRINT "J=";J
120       B(I,J)=0
140     NEXT J
141     PRINT "INITIALIZED ",I,J
150   NEXT I
151   !
160 Input_1:  ! enter data
161   !
170   INPUT "ENTER RADIUS OF CIRCLE",R
180   IF R>0 THEN Compute
190   BEEP
200   PRINT R;" IS NOT A VALID RADIUS ENTRY"
210   GOTO Input_1
220   !
230 Compute:  ! continue
240   !
250   C=FNCirc(R)
260   CALL Print(C,R)
270   PRINT "END"
280   END
281   !
290   DEF FNCirc(R)
300     C=2*R*PI
310     RETURN C
320   FNEND
321   !
330   SUB Print(C,R)
340     PRINT "RADIUS     CIRCUMFERENCE"
350     PRINT R,C
360   SUBEND
380   !
```

Figure 4. Example of Structured ROM indentation.

4.3 PROGRAMMING CONVENTIONS

4.3.1 Naming

Naming conventions shall be uniform throughout the SNAP software. For example, don't call the same temperature "Temp" in one program segment and "T" in another. Use meaningful names for program segment labels, variable names, and file names. Avoid acronyms not in general use. Avoid using the characters zero and one in names unless they are part of an identifying number. To simplify SNAP to TESS program conversions, limit names to six characters or less.

Names may be assigned using the implicit FORTRAN convention, (i.e., starting integer and character variables with one of the letters I-N). This convention reduces the need to refer to the explicit variable declarations at the beginning of the program segment. A comment statement shall be used to identify programs using this implicit naming convention.

4.3.2 Numerical

4.3.2.1 Constants

Constants shall be defined rather than calculated (e.g., use Half=0.5 instead of Half=1/2). Each constant shall have at least one more significant digit than any variable used with that constant. For example, if a variable is entered with 0.01 precision, then any constants used with that variable shall be defined to 0.001 precision.

4.3.2.2 Mixed-Mode Expression

To simplify SNAP to TESS program conversions, mixed-mode expressions should be avoided. Note that the intrinsic integer (INT) function in HP-BASIC does not truncate as does the INT function in FORTRAN.

4.3.2.3 Grouping

The 9845 automatically deletes unneeded blanks and parentheses from arithmetic operations. Comment statements shall be used to clarify the order of evaluation of complex operations.

4.3.2.4 Significant Digits

The number of significant digits of the output shall not exceed the number of significant digits of the input. Rounding by the programmer shall not occur until the final computational step.

SHORT precision variables and calculations shall be used only when computer memory is exhausted and program overlays cannot be used. In all cases, the accuracy requirements of the PPS must be met.

4.3.3 Narrative Description

SNAP programs should be internally documented using comments in the source code. Some programs may exceed the memory capacity of the computer or have unacceptably long execution times when documented to the extent described below. With the prior approval of the NEPRF SNAP principal investigator, such programs shall be delivered in two versions:

1. A fully documented but non-executable version.
2. An executable version stripped of comments.

4.3.3.1 Abstracts

Each program and subprogram shall include at the beginning of the executable coding a textual description of its inputs; outputs; purpose; a list of other segments called; and a list of all calling segments.

An alphabetized table of variable names, variable definitions and the range of allowable values shall be included in the abstract.

The initial abstract of the main program segment shall also include the name of the activity or company which developed the program, the name and telephone number of a point of contact, and the date of the last program modification.

The initial abstract shall also explain the convention used for latitude and longitude if applicable. For example, degrees and minutes versus decimal degrees; positive and negative degrees versus north and south labels; latitudes varying from 0 to 90 versus 0 to 180; and longitudes varying from 0 to 180 versus 0 to 360. Figure 5 is an example of an initial abstract.

4.3.3.2 Comments

Comments shall be used throughout the code. Comments may be grouped into a single block preceding a group of source statements which perform a related procedure. Use ! instead of REM for comments. Blank lines used for spacing should also use an !.

4.3.4 Labels

HP-BASIC allows program lines to be labeled with a unique name. Without the cross reference capability of the Structured Programming ROM, however, locating these labels in a long program can be time consuming. For readability, it is more efficient to use line numbers than labels. A composite method of using labels and line numbers follows:

```
30   GO TO 1130 ! Set_up
       .
       .
       .

1130 Set_up: ! initialize all variables

       .
       .
       .
```

This lets the programmer use labels until the program is finished. Then line numbers are inserted and the labels are left in place to aid program readability. The use of line labels versus line numbers is left to the programmer since NEPRF has a 9845B-275 with cross reference capability.

```
10  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
20  ! 23 JUN 1983                                                            !
30  !                     D-VALUE PROGRAM (SNAP)                             !
40  !                                                                        !
50  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
60  !                                                                        !
70  !         INTERACTIVE PROGRAM TO CALCULATE D-VALUES FROM                 !
80  !         RADIOSONDE OBSERVATIONS OF PRESSURE AND HEIGHT.                !
90  !         THE D-VALUE IS THE DIFFERENCE, Dval,                           !
100 !         BETWEEN THE OBSERVED HEIGHT, Z ABOVE MEAN SEA                  !
110 !         LEVEL, OF A PARTICULAR PRESSURE SURFACE, AND                   !
120 !         THE PRESSURE ALTITUDE, Zst (THE HEIGHT OF THE                  !
130 !         SAME PRESSURE SURFACE IN THE U.S. STANDARD                     !
140 !         ATMOSPHERE), I.E.  Dval=Z-Zst.                                 !
150 !                                                                        !
160 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
170 !                                                                        !
180 !          DEVELOPED BY THE NAVAL ENVIRONMENTAL PREDICTION               !
190 !          RESEARCH FACILITY (NEPRF), MONTEREY, CA 93943.                !
200 !          TELEPHONE (408) 646-2937   AUTOVON-878-2837.                  !
210 !          CONTACTS: MR. TERRY BROWN OR MR. SAM BRAND.                   !
220 !                                                                        !
230 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
240 !                                                                        !
250 !          INPUTS:                                                       !
260 !                  LATITUDE AND LONGITUDE OF OBSERVED DATA               !
270 !                  DATE AND TIME OF OBSERVED DATA                        !
280 !                  OBSERVED PRESSURES AND HEIGHTS                        !
290 !                  HEIGHT INTERVAL AT WHICH D-VALUES ARE CALCULATED      !
300 !                  UNITS OF HEIGHT AND D-VALUE                           !
310 !                                                                        !
320 !          OUTPUTS:                                                      !
330 !                  SPECIFIED HEIGHTS AND CORRESPONDING D-VALUES          !
340 !                                                                        !
350 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
360 !                                                                        !
370 !                      LIMITS OF ALGORITHM                               !
380 !                                                                        !
390 !          MAXIMUM OF 50 LEVELS OF DATA MAY BE ENTERED.                  !
400 !          MAXIMUM HEIGHT IS 11,000 METERS.                              !
410 !          ALL PRESSURES MUST BE IN MILLIBARS.                           !
420 !          HEIGHTS MUST BE EITHER FEET OR METERS.                        !
430 !          NO ERROR CHECKS ARE MADE OF LOCATION, DATE, OR TIME ENTRIES.  !
440 !                                                                        !
450 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
460 !                                                                        !
470 !                                                                        !
480 !                      SYNOPSIS OF SUBPROGRAMS                           !
490 !                                                                        !
500 !          Eloc: ENTER STATION LOCATION DATA                             !
510 !          Edti: ENTER DATE AND TIME DATA                                !
520 !          Einc: ENTER ALTITUDE INCREMENT DATA                           !
530 !          Eobs: ENTER RADIOSONDE PRESSURE AND HEIGHT DATA               !
540 !          Sort: SORT DATA INTO DESCENDING ORDER OF PRESSURE             !
550 !          Chck: CHECK PRESSURE AND HEIGHT DATA FOR ERRORS               !
560 !          Pdat: PRINT TABLE OF PRESSURE AND HEIGHT DATA                 !
570 !          Corr: ALLOW OPERATOR TO CORRECT DATA                          !
580 !          Chng: ALLOW OPERATOR TO CHANGE EXISTING ENTRY                 !
590 !          Delt: ALLOW OPERATOR TO DELETE EXISTING ENTRY                 !
600 !          Addl: ALLOW OPERATOR TO ADD OR INSERT AN ENTRY                !
610 !          Prnt: PRINT HEADING FOR D-VALUE TABLE                         !
620 !          Calc: CALCULATE D-VALUES AND PRINT RESULTS                    !
630 !                                                                        !
640 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
650 !                                                                        !
660 !          VARIABLES IN THIS PROGRAM SEGMENT       ALLOWABLE VALUES      !
670 !                                                                        !
680 !            Cue$     OPERATOR RESPONSE TO QUERY          Y OR N         !
690 !            Dinc     D-VALUE INCREMENT                    > 0           !
700 !            Ierr     ERROR INDICATOR                     0 TO 3         !
710 !            Lev      NUMBER OF LEVELS OF DATA            1 TO 50        !
720 !            Loc$     LOCATION OF OBSERVATION          ANY CHARACTERS    !
730 !            Pob      OBSERVED PRESSURE DATA               > 0           !
740 !            Time$    DATE AND TIME OF OBSERVATION     ANY CHARACTERS    !
750 !            Unit$    UNITS OF ALTITUDE INCREMENT          F OR M        !
760 !            Zob      OBSERVED HEIGHT DATA                 > 0           !
770 !            Zunit$   UNITS OF OBSERVED HEIGHTS            F OR M        !
780 !                                                                        !
790 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Figure 5. Example of an abstract at the beginning of of a program.

4.3.5 Program Control Statements

4.3.5.1 FOR...NEXT Loops

Do not exit and re-enter the range of a loop. Do not perform invariant calculations and assignments within loops.

4.3.5.2 IF Statements

Do not make equality tests (=) on floating point data unless exact equality is desired. In general, use a tolerance test. For example,

```
IF (NOT(ABS(A-B)<=Tolrnc)) THEN ...
```

4.3.6 Specification and Data Statements

Explicitly specify the OPTION BASE (usually 1) at the beginning of the program.

Variable types, (i.e., INTEGER, REAL), shall be explicitly declared at the start of each program segment and in COMMON statements. Character variables shall be declared INTEGER. Dimension all arrays in these declaration statements. Document with comments any redimensioned arrays. Alphabetize variable names within declaration statements. Bracket declarations with blank lines. For example,

```
!
COM REAL Altim, Dval(100), INTEGER Count
INTEGER I, J, Day(31)
INTEGER Month$(12)
REAL Humid, Press, Temp(2,10)
!
```

The 9845 computer performs all operations using full-precision (REAL) accuracy. The conversion to and from an INTEGER variable type slows program execution. Another problem may occur when inverting a matrix that is not full precision since rounding errors will affect the accuracy of the results. If a program has an unacceptably long execution time or is inaccurate due to rounding errors (as determined by the NEPRF SNAP principal investigator), then all explicit declarations shall be preceded by an ! and remain in the code as comments.

4.3.6.1 Labeled COMMON

HP-BASIC does not provide for labeled COMMON. A comment shall be used to distinguish between different COMMON blocks. For example,

```
! COMMON/TEMPS/
COM REAL Airtmp, Seatmp, Wetblb
!
! COMMON/WINDS/
COM REAL Wnddir, Wndspd
```

Each copy of the "labeled" COMMON shall be the same length and contain the same variable names.

4.3.6.2 DATA Statements

Comment statements shall be used to delineate related groups of DATA statements. The arrangement of information that is part of an array shall correspond to the array dimensions. For example,

```
!
REAL Airtmp(3,4)
     .
     .
     .
!
!  Air temps celsius
!
DATA  -15.8,-12.3,-8.0,-5.3
DATA  0.4,3.5,12.4,15.0
DATA  21.9,23.0,22.5,24.7
!
```

Information that is not part of an array shall be arranged so that each DATA statement contains: <5; 5; or 10 items. For example,

```
!
DATA 1,2,3,4,5,6,7,8,9,10
DATA 11,12,13,14,15,16,17,18,19,20
DATA 21,22,23,24,25
DATA 26,27
!
```

4.3.7 Subprograms

HP-BASIC has two types of subprograms. Subroutine subprograms (CALL syntax) and Function subprograms (FN syntax). It also has subroutines (GOSUB syntax). Subprograms are preferable to subroutines due to their similarity to FORTRAN syntax. Multiple points to which the program may return from a subprogram shall not be allowed.

A FUNCTION shall be used only for its returned value, e.g., do not modify global data or arguments within a FUNCTION.

Data in pass parameter lists shall be ordered so that "given" data are first, "both" (given and yielded) data are next, and "yielded" data are last. Alphabetize parameters within their respective grouping. For example,

```
CALL Demo(Constant,Given,Both,Result,Yielded)
```

4.3.8 Abnormal Termination

System errors (see glossary) shall not abort program execution. These errors shall be trapped using the ON ERROR syntax which shall transfer program execution to a centralized error routine. A maximum of one STOP statement is allowed in each SNAP program and it must reside in this error routine. This routine shall automatically print a hard copy message similar to figure 6.

```
------------------------------------------------------------------------

Forward the following message
along with a copy of your data
if possible to:

                    NEPRF
                    Attn: SNAP WORK UNIT 6.2-34
                    MONTEREY,  CA  93943

------------------------------------------------------------------------

↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓ MESSAGE ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓

                    THIS IS ↓NPL↓ ICING PROBABILITY Code
                    LAST MOD: (Nov. 1983)
                    PROGRAM: INPUTS
                    OPTION:  INPUT RADIOSONDE DATA SET
                    ERROR MESSAGE: ERROR 18 IN LINE 3090

↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓ END OF MESSAGE ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓

------------------------------------------------------------------------
```

Figure 6. Example of an abnormal termination message.

4.3.9 Miscellaneous

Calculations shall use the MKS unit system as specified for TESS programs. Input/output may be in other unit systems if applicable.

Explicitly declare the units to be used in trigonometric calculations (i.e., DEG, RAD, or GRAD) at the beginning of the program. The preferred units are RADs which are used in FORTRAN.

Use only simple integers as indices and subscripts. These indices shall be explicitly declared as INTEGER variables.

Do not use multiple statements (i.e., no X=Y=Z=3.14).

Initialize every variable, usually with the applicable default value, before using.

Do not depend on the values of "local" variables computed on a previous call to a routine.

Do not use LET in assignment statements.

End each program segment with a line of !'s and the words END SEGMENT NAME. Begin each program segment with a line of !'s and the words BEGIN SEGMENT NAME. For example:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! END ANALYSIS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! BEGIN FORECAST
```

Explicitly declare MASS STORAGE IS (usually ":T15") at the beginning of the program.

Use Y and N for replies to Yes/No queries. The special function keys described below can be used.

4.3.10 Special Function Keys

The 9845 Special Function Keys (SFK's) are marked k0 through k15. These SFK's can be defined by the programmer to perform operations with a single keystroke. The use of SFK's is encouraged. To maintain consistency with existing SNAPs and the Integrated Refractive Effects Prediction System (IREPS) program, the SFK definitions shown in figure 7 should be observed. Changes or additions shall be documented in the program abstract.

```
KEY 0
  -Clear line
   BACK-UP
  -Continue

KEY 1
  -Clear line
   CONT Option
  -Execute

KEY 2
  -Clear line
   EXIT GRAPHICS
  -Execute

KEY 3
  -Clear line
   GRAPHICS
  -Execute

KEY 4
  -Clear line
   Prints$="PRINTER OFF"
  -Execute

KEY 5
  -Clear line
   Prints$="PRINTER ON"
  -Execute

KEY 6
  -Clear line
   Y
  -Continue

KEY 7
  -Clear line
   N
  -Continue

KEY 8
  -Clear line
   CONT Repeat
  -Execute

KEY 9
  -Clear line
   LOAD "AUTOST",1
  -Execute
```

Figure 7. Existing Special Function Key definitions.

KEY 0 implements a "back up one step" procedure which allows the user to return to the immediately preceding prompt. A program segment using this feature is:

```
100 Date:   ! enter date
110    Reply$="TODAY"
120    INPUT "ENTER DATE",Reply$
130    IF Reply$="BACK-UP" THEN Start
140    Date$=Reply$
150    !
160 Lat:    ! enter latitude
170    Reply$="NOT ENTERED"
180    INPUT "ENTER LATITUDE",Reply$
190    IF Reply$="BACK-UP" THEN Date
200    Lat$=Reply$
210    !
220 Lon:    ! enter longitude
230    Reply$="NOT ENTERED"
240    INPUT "ENTER LONGITUDE",Reply$
250    IF Reply$="BACK-UP" THEN Lat
260    Lon$=Reply$
270    !
```

KEY 1 transfers execution to the program line labeled "Option". Usually this label marks the beginning of the program segment which displays the main menu. Thus, the user can return at any time to the main menu.

KEY 2 allows printed output to be displayed by EXITing GRAPHICS. This is more applicable to the 9845A which does not allow simultaneous display of text and graphics as does the 9845B-275.

KEY 3 allows plotted output to be displayed by enabling the GRAPHICS capability. It is the reverse of KEY 2.

KEY 4 and KEY 5 allow the user to specify the CRT or the internal printer as the output device by assigning the variable named Print$ a value which is checked by the program. The appropriate PRINTER IS statement is then executed. For example,

```
IF Print$="PRINTER ON" THEN PRINTER IS 0
```

or

```
IF Print$="PRINTER OFF" THEN PRINTER IS 16
```

KEY 6 and KEY 7 let the user answer yes/no questions with one keystroke instead of two.

KEY 8 is used in the IREPS program to repeat the last output. It is not applicable to other programs.

KEY 9 re-starts the entire program run. If your program does not use the AUTOSTart capability of the 9845, you must change the word "AUTOST" to the name of the first program you want executed.

4.3.11 Program Overlays

Overlays (see glossary) shall be used when needed to ensure that each SNAP can be loaded into the memory of either a 9845A or 9845B. SNAPs written specifically for the 9845B-275 are exempt from this requirement.

When a program is divided into overlays, the program should display a message indicating which overlay is currently executing.

4.3.12 Default Values

The ability to select default values for user entries as specified in the PPS shall be included in each SNAP. Default values shall be displayed to the user and be selected by simply pressing the CONT key. Frequently, defaults represent suggested values for data that is difficult for SNAP users to obtain.

The use of defaults is suggested even when not specified in the PPS. For example, whenever the user is presented a list of choices, the default should be the first choice in the list. This follows the convention of the IREPS program. It follows that the default for Yes/No questions should be a Yes (Y).

5. HUMAN-FACTORS STYLE GUIDE

Consider the following human-factors design principles while writing SNAPs:

- Provide immediate and obvious feedback to the user. Echo all user entries.

- Be consisten . For example, place all error messages at the same screen location.

- Minimize user memory demands. Display default input values to the user. Menu displays help the user make choices.

- Simplify. Use short prompts and incorporate default values into the program whenever possible. Limit the number of choices the user must consider at any one time.

- Match the program to the user's skill level. Consider what the user is expected to do, what decisions need to be made and what information is needed to make those decisions.

- Tell the user where he is and how to get back to where he came from. Menu-driven programs do this by providing a main menu which serves as a home base. The program begins with this main menu from which the user can select various options and then return.

- Tell the user what is happening. Display screens should be titled: for example, "WIND PROBABILITY CALCULATION PROCEDURE". If a section of the program takes more than 5 seconds to execute, tell the user how long he must wait. For example, "Now calculating wind probability isopleths: time required is 2 minutes". The user should be advised of the program's progress during these longer operations. For example, "30 knot probabilities completed. Now doing 50 knot probabilities". These updates should be displayed at about 30 second intervals.

5.1 DATA ENTRY

The following guidelines apply to programs in which the user enters information through the keyboard. (The 9845 keyboard is described in Appendix A). The data entry process should typically consist of the following sequence of steps:

- program displays a prompt
- user enters information
- program echoes the entry on the screen
- program error checks the entry
- program displays an error message if appropriate
- information is edited as needed
- information is accepted

5.1.1 Prompting

The program should provide a prompt for every data input. Prompts should appear in the Display Line (line 22) of the CRT.

The entry format should be displayed in inverse video and precede the instruction to the user. This position allows the user to see his entry displayed directly below the format. Generally, use the character "#" to show formats for digits and the letter "C" to show formats for alphanumeric entries. More specific formats can be used. For example, the format for a calendar date entry might be "MMDDYY" for Month, Day, and Year. Established formats, such as the World Meteorological Organization (WMO) radiosonde code, may be used. For example, the WMO code uses "ddfff" to format wind direction and speed reports.

Clarifying instructions and entry examples, if used, should be in parentheses immediately following the user instruction.

Range limits should be next in the prompt line in braces ({...}).

Default values should be at the end of the prompt line in brackets ([...]). For example,

###.# Enter Temperature (Celsius), {-50 to 50}, [10].

where:

###.#	is the entry format showing that 0.1 degree precision is expected and that the field length is five places including sign and decimal point.
Enter Temperature	is the instruction to the user.
(Celsius)	is a clarifying instruction.
{-50 to 50}	is the range of acceptable values.
[10]	is the default value obtained by simply pressing the CONT key.

Some entries may require more explicit instructions than can be fit in the Display Line. In such cases, the explanatory information should be displayed above the Display Line in the Printout Area. The explanatory information should be cleared from the screen once the data entry has been completed. The prompt itself should still follow the guidance above.

If there are similar or identical data entry requirements in different parts of the program, prompt consistently.

5.1.2 Entering Data

Minimize the length of the information to be entered. For example, don't make the user enter "SEPTEMBER" when the abbreviation "SEP" will suffice. If data consist of logically related groups (e.g., date-time group), permit the user to enter several fields together instead of separately. For example, don't make the user enter the day in response to one prompt, the month in response to a second prompt, and the year in response to a third prompt. Instead, let the user enter the day, month, and year at the same time in response to a single prompt.

Data shall be entered in the appropriate units. For example, since tropical cyclone warnings give wind speeds in knots, the user shall enter wind speed data in knots. Conversion to other units shall be done by the program and be documented with comments.

The program should accept both upper and lower case letters interchangeably, unless differentiation is required. For example, the user should be able to enter either a "Y" or a "y" in order to affirm a Yes/No question.

5.1.3 Error Check

Check all entries for errors. Anticipate possible errors such as entering a letter in place of a number. Check all entries against range and length limits. Catch entries which cause illegal program actions such as division by zero.

When errors are detected, alert the user, identify the error, and tell him how to recover. It is not enough to beep or display the word "ERROR". Alerting signals must differ from the normal background. Using beeps and flashing messages throughout the program will reduce their value in error routines. Consider reserving beeps and flashing messages for errors.

Error messages shall be placed consistently at the same screen location, just above the Display Line is a good place. The message shall show the erroneous entry. The message shall tell what is wrong -- for example, that the entry is too long -- and what to do -- for example, abbreviate month with three letters. An example of a helpful error message is shown below.

ERROR. Date 2/31/83 contains invalid day.

MM/DD/YY Enter date.

Some errors are more serious than others. For example, a program that allows a user to purge files with a single keystroke is a disaster waiting to happen. Protect the user in such cases by making data entry require two stages. When the user makes a potentially disastrous entry, display a new message describing the consequences. The user should be given a chance to continue or back out. Such a display is shown below.

** WARNING **

You have selected the "PURGE FILES" option.
Your data files will be purged if you continue.

C ABORT the PURGE FILES option?, (Y/N), [Y]

5.1.4 Editing

Permit the user to edit entries before the program accepts data for calculations. In some programs, data should be edited at more than one stage: during initial entry, after a block of related entries have been made, and after the data have become part of a database (if applicable).

The 9845 automatically allows editing during initial entry. The user's entry is displayed as it is typed in the Keyboard Entry Area (lines 23-24) of the CRT. It can be edited using the display and character editing keys until the user presses the CONT key. In addition, a "back up one step" procedure can be included in each program to allow the user to return to the immediately preceding prompt and re-enter the data. This feature can be programmed into Special Function Key 0 as described in section 4.3.10. When the user sees that the echo of his entry is a mistake, the backup utility offers a quick fix.

This back up utility doesn't help when the user does not see the mistake until several other entries have been made. In block editing, the program displays a related group of entries and also displays a prompt asking if the user wants to edit any of them. If the user does want to edit, he defines the entry to be changed, usually by entering a line or index number. The user is prompted to re-enter the indicated data. When the user indicates that no more changes are needed, the program moves on to the next step.

Editing a database can be done with a separate utility program or as a part of the SNAP itself. In some SNAPs, data entered and stored on tape during a previous program execution are read and displayed for editing before continuing with a new execution.

5.2 DISPLAY SCREEN DESIGN

Users generally find it easier to read stationary pages than moving pages. The 9845 will begin scrolling once the 20 Printout Area lines of the screen are filled. (The 9845 CRT display screen is described in Appendix A). Instead of scrolling, it is preferable to clear the screen before putting up a new display.

Most displays should be titled at the top-center of the screen. In addition to a title, most displays will contain other information such as entry echoes, prompt line, error-messages, etc. A separate screen area should be allocated for each information category. Information should not stray from its assigned area. Try to separate each area of the screen from the next with rows or columns of blank spaces or with lines.

Determine what information the user needs at each point in the program and display only that. Each screen should convey one logically connected thought or step.

Follow prevailing conventions. Numeric information usually should be displayed in tabular format, i.e., beneath column headings from top to bottom. If the user is entering data from a standardized form, the display should mimic the layout of that form.

Data shall be displayed in the appropriate units as specified in the PPS. Usually these are the same units used for entering input.

Display information in a recognizable order. Long lists should be shown in an order familiar to the user. For example, alphabetic, numeric, or chronological order. In menus, the most frequently selected options should be listed at the top and the infrequent selections should be at the bottom.

Text should be left justified. Numeric information normally should be right justified. Where the number of decimal places varies on successive lines, decimal points on all lines should align at a particular tab value.

5.3 MENU-DRIVEN PROGRAMS

SNAPs should generally be menu-driven programs since menus make few demands on human memory, are easy to learn, and always show the user the possible paths to follow.

The menu should have three parts: title, list of menu choices, and a prompt line. The menu title should be centered at the top of the display and should end with the word "menu". The prompt generally should be displayed in the default 9845 Display Line at the bottom of the screen. The prompt should be brief. The following is a sample menu.

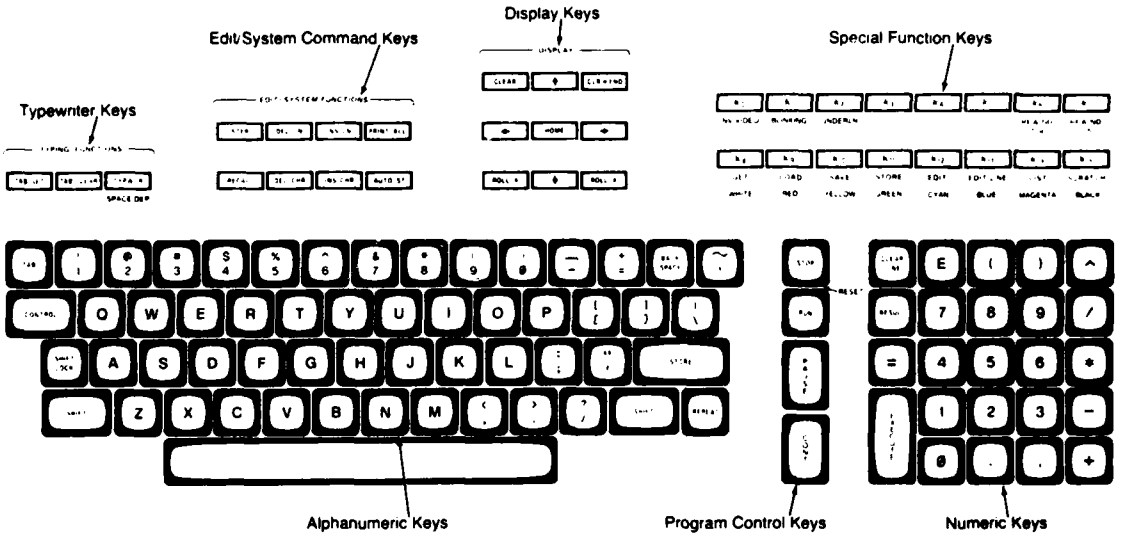
```
                    MAIN MENU

     Select one of the following actions by entering its number


               1.  ENTER DATA
               2.  GENERATE REPORT
               3.  EDIT DATA
               4.  START NEW FILE
               5.  PURGE FILE
               6.  QUIT

     #  Enter Choice,  {1-6},  [1].
```

Menu selection should be done by entering a number. The default should be the first choice. The 9845A does not allow cursor selection. Programs written solely for the 9845B-275 may use cursor selection. When using numbers in menus, avoid their use in nonmenu displays. If you are displaying a list of instructions, for example, precede each instruction by a hyphen instead of a number. Confusion can be reduced by titling menus as menus and titling other displays appropriately.

APPENDIX A

A-1. 9845 KEYBOARD

The layout of the 9845 keyboard is shown below.



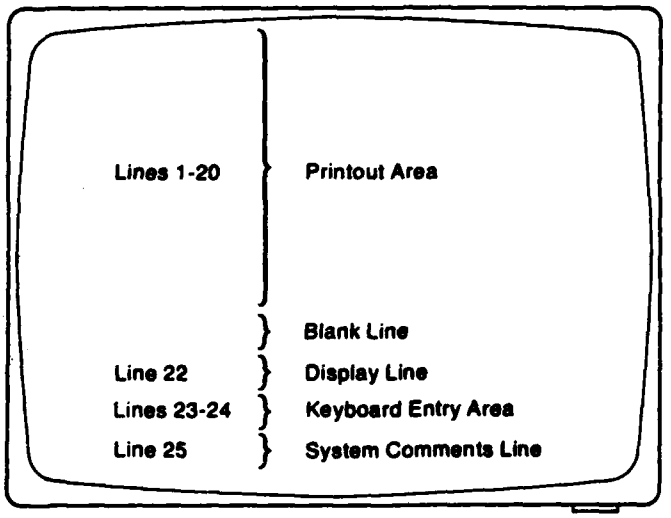
A-2. 9845 CRT DISPLAY SCREEN

The CRT is divided into several areas:

- Printout area (lines 1-20) is used for:
  - > echoing user entries
  - > instructions and background information
  - > tabular displays and menus
  - > error messages
  - > other messages
- Display line (22) is used for:
  - > input prompts
  - > short messages generated by DISP statement
- Keyboard entry area (lines 23-24) is used for:
  - > user entries
- System comments line (25) is used for:
  - > system error messages and indicators

The 9845B option 275 also has a CRT area for soft key labels. Programs written solely for the 9845B-275 may use this capability.

The CRT display screen is shown below.



APPENDIX B

GLOSSARY

Abstract	A condensed description or summary of a computer program.
Calling Program	When a subprogram is being executed, the program segment (either main program or subprogram) which calls the subprogram is the calling program. Control returns to the calling program when the called program is completed.
Construction	A group of logically related BASIC statements.
Control Structure	Another term for a construction.
Default Value	The value assigned to a variable when the user does not enter any information in response to a prompt but simply presses the CONT key.
Function Subprogram	A program segment that returns a single numeric or character string value to the calling program. It is similar to a FORTRAN FUNCTION.
Main Program	The part of a program from which subprograms can be called. Subprograms cannot call the main program.
MKS Unit System	Unit system using meters for length, kilograms for mass, seconds for time, and kelvins for temperature. Other acceptable units are Celsius degrees for temperature and millibars for pressure.
Operand	A quantity entering into or arising from an operation; it may be an argument, result, parameter, or the location of the next instruction.
Program	The complete sequence of coded instructions and data necessary to solve a problem.
Program Segment	An independent group of program statements. The main program and each subprogram are program segments.

Structured Programming	A programming technique using the following concepts: (1) Program uses only the three control structures of a simple sequence of operations; a conditional branch; and a repetition of an operation. (2) Program uses limited or no "GO TO" logic. (3) Program is divided into constituent parts and these parts are divided into their constituents. This breakdown continues until a level is reached where there is no subservient level. Lower levels do not call higher levels.
Subprogram	A group of statements that performs a certain task under the control of the calling program segment. It differs from an HP-BASIC subroutine (GOSUB...RETURN syntax) since it is a seperate segment which comes after the main program.
Subroutine	A group of HP-BASIC statements which can be accessed from different places in a program segment. The subroutine is part of that program segment.
Subroutine Subprogram	A program segment that performs a specific task but is not part of the calling program. It is similar to a FORTRAN SUBROUTINE and can return more than one (or none) numeric or string values.
System Error	An error detected by the 9845 operating system which normally causes program execution to cease. These errors are numbered and an error message is automatically displayed which identifies the error type.

APPENDIX C

Some Simulated Structured Programming Constructions in HP - BASIC

1. IF...THEN single statement

```
100 IF condition THEN statement
```

2. IF...THEN multiple statement

```
 80 IF NOT condition THEN GOTO 100
  .     statements
  .
  .
100 ! END IF
```

3. IF...THEN...ELSE

```
 20 IF NOT condition THEN GOTO 50
  .     statements-1
  .
  .
 40 GOTO 100
 50 ! ELSE
  .     statements-2
  .
  .
100 ! END IF
```

4. Multiway selection, ELSE IF

```
 20 IF NOT condition-1 THEN GOTO 50
  .     statments-1
  .
  .
 40 GOTO 150
 50 ! ELSE
 60 IF NOT condition-2 THEN GOTO 90
  .     statements-2
  .
  .
 80 GOTO 150
 90 ! ELSE
100 IF NOT condition-3 THEN GOTO 130
  .     statements-3
  .
  .
120 GOTO 150
130 ! ELSE
  .     statements-4
  .
  .
150 ! END IF
```

5. Multiway selection, CASE

```
20 IF variable < 1 THEN GOTO 140
30 IF variable > 3 THEN GOTO 140
40 ON variable GOTO 50, 80, 110
50 ! CASE 1
 .     statements-1
 .
 .
70 GOTO 160
80 ! CASE 2
 .     statements-2
 .
 .
100 GOTO 160
110 ! CASE 3
 .     statements-3
 .
 .
130 GOTO 160
140 ! ELSE
 .     statements
 .
 .
160 ! END CASE
```

6. DO WHILE repetition

```
20 IF NOT condition THEN GOTO 60
 .     statements
 .
 .
50 GOTO 20
60 ! END WHILE
```

7. DO UNTIL or REPEAT repetition

```
20 ! REPEAT
 .
 .
 .
50 IF NOT condition THEN GOTO 20
```

8. DO FOR repetition

```
20 FOR I = L TO M STEP N
 .     statements
 .
 .
50 NEXT I
```

Note: If a program using these contructions is stripped of comments by a comment deleting utility program, then the destination lines listed in the GOTO statements will be deleted and the program will not run. The programmer may need to use line labels to avoid this problem. For example, instead of the syntax shown in example 3 above, use:

```
20 IF NOT condition THEN GOTO Line_50
 .      statements-1
 .
 .
40 GOTO Line_100
50 Line_50:  ! ELSE
 .     statements-2
 .
 .
100 Line_100:  !END IF
```

DISTRIBUTION

COMMANDER IN CHIEF
U.S. ATLANTIC FLEET
ATTN: FLT METEOROLOGIST
NORFOLK, VA 23511

COMMANDER IN CHIEF
U.S. ATLANTIC FLEET
ATTN: NSAP SCIENCE ADVISOR
NORFOLK, VA 23511

COMMANDER IN CHIEF
U.S. NAVAL FORCES, EUROPE
ATTN: METEOROLOGICAL OFFICER
FPO NEW YORK 09510

CINCUSNAVEUR
ATTN: NSAP SCIENCE ADVISOR
BOX 100
FPO NEW YORK 09501

COMMANDER SECOND FLEET
ATTN: METEOROLOGICAL OFFICER
FPO NEW YORK 09501

COMSECONDFLT
ATTN: NSAP SCIENCE ADVISOR
FPO NEW YORK 09501

COMTHIRDFLT
ATTN: FLT METEOROLOGIST
PEARL HARBOR, HI 96860

COMSEVENTHFLT
ATTN: FLT METEOROLOGIST
FPO SAN FRANCISCO 96601

COMTHIRDFLT
ATTN: NSAP SCIENCE ADVISOR
PEARL HARBOR, HI 96860

COMSEVENTHFLT
ATTN: NSAP SCIENCE ADVISOR
BOX 167
FPO SEATTLE 98762

COMSIXTHFLT
ATTN: FLT METEOROLOGIST
FPO NEW YORK 09501

COMSIXTHFLT/COMFAIRMED
ATTN: NSAP SCIENCE ADVISOR
FPO NEW YORK 09501

COMMANDER NAVAL AIR FORCE
U.S. ATLANTIC FLEET
ATTN: NSAP SCIENCE ADVISOR
NORFOLK, VA 23511

COMNAVAIRPAC
ATTN: NSAP SCIENCE ADVISOR
NAS, NORTH ISLAND
SAN DIEGO, CA 92135

COMNAVSURFLANT
ATTN: NSAP SCIENCE ADVISOR
NORFOLK, VA 23511

COMNAVSURFPAC
(OO5/N6N)
ATTN: NSAP SCIENCE ADVISOR
SAN DIEGO, CA 92155

COMMANDER
MINE WARFARE COMMAND
ATTN: NSAP SCIENCE ADVISOR
CODE 007
CHARLESTON, SC 29408

COMMANDER
AMPHIBIOUS GROUP 2
ATTN: METEOROLOGICAL OFFICER
FPO NEW YORK 09501

COMMANDER
AMPHIBIOUS GROUP 1
ATTN: METEOROLOGICAL OFFICER
FPO SAN FRANCISCO 96601

COMMANDER
OPTEVFOR
NAVAL BASE
NORFOLK, VA 23511

COMMANDER
OPTEVFOR
ATTN: NSAP SCIENCE ADVISOR
NORFOLK, VA 23511

OFFICER IN CHARGE
OPTEVFOR, SUNNYVALE
NAVAL AIR STATION
MOFFETT FIELD, CA 94035

COMMANDING OFFICER
USS AMERICA (CV-66)
ATTN: MET. OFFICER, OA DIV.
FPO NEW YORK 09531

COMMANDING OFFICER
USS FORRESTAL (CV-59)
ATTN: MET. OFFICER, OA DIV.
FPO MIAMI 34080

COMMANDING OFFICER
USS INDEPENDENCE (CV-62)
ATTN: MET. OFFICER, OA DIV.
FPO NEW YORK 09537

COMMANDING OFFICER
USS J. F. KENNEDY (CV-67)
ATTN: MET. OFFICER, OA DIV.
FPO NEW YORK 09538

COMMANDING OFFICER
USS NIMITZ (CVN-68)
ATTN: MET. OFFICER, OA DIV.
FPO NEW YORK 09542

COMMANDING OFFICER
USS D. D. EISENHOWER (CVN-69)
ATTN: MET. OFFICER, OA DIV.
FPO NEW YORK 09532

COMMANDING OFFICER
USS SARATOGA (CV-60)
ATTN: MET. OFFICER, OA DIV
FPO MIAMI 34078

COMMANDING OFFICER
USS CONSTELLATION (CV-64)
ATTN: MET. OFFICER, OA DIV
FPO SAN FRANCISCO 96635

COMMANDING OFFICER
USS CORAL SEA (CV-43)
ATTN: MET. OFFICER, OA DIV.
FPO NEW YORK 09550

COMMANDING OFFICER
USS ENTERPRISE (CVN-65)
ATTN: MET. OFFICER, OA DIV.
FPO SAN FRANCISCO 96636

COMMANDING OFFICER
USS KITTY HAWK (CV-63)
ATTN: MET. OFFICER, OA DIV.
FPO SAN FRANCISCO 96634

COMMANDING OFFICER
USS MIDWAY (CV-41)
ATTN: MET. OFFICER, OA DIV.
FPO SAN FRANCISCO 96631

COMMANDING OFFICER
USS RANGER (CV-61)
ATTN: MET. OFFICER, OA DIV.
FPO SAN FRANCISCO 96633

COMMANDING OFFICER
USS CARL VINSON (CVN-70)
ATTN: MET. OFFICER, OA DIV.
FPO SAN FRANCISCO 96629

COMMANDING OFFICER
USS NEW JERSEY (BB-62)
FPO SAN FRANCISCO 96688

PCO, IOWA (BB-61)
SUPERVISOR OF SHIPBUILDING
CONVERSION & REPAIR, USN
PASCAGOULA, MS 39567

COMMANDING OFFICER
USS MOUNT WHITNEY (LCC-20)
ATTN: MET. OFFICER
FPO NEW YORK 09517

COMMANDING OFFICER
USS BLUERIDGE (LCC-19)
ATTN: MET. OFFICER
FPO SAN FRANCISCO 96628

COMMANDING OFFICER
USS GUADALCANAL (LPH-7)
ATTN: MET. OFFICER
FPO NEW YORK 09562

COMMANDING OFFICER
USS GUAM (LPH-9)
ATTN: MET. OFFICER
FPO NEW YORK 09563

COMMANDING OFFICER
USS INCHON (LPH-12)
ATTN: MET. OFFICER
FPO NEW YORK 09529

COMMANDING OFFICER
USS IWO JIMA (LPH-2)
ATTN: MET. OFFICER
FPO NEW YORK 09561

COMMANDING OFFICER
USS NASSAU (LHA-4)
ATTN: MET. OFFICER
FPO NEW YORK 09557

COMMANDING OFFICER
USS SAIPAN (LHA-2)
ATTN: MET. OFFICER
FPO NEW YORK 09549

COMMANDING OFFICER
USS NEW ORLEANS (LPH-11)
ATTN: MET. OFFICER
FPO SAN FRANCISCO 96627

COMMANDING OFFICER
USS OKINAWA (LPH-3)
ATTN: MET. OFFICER
FPO SAN FRANCISCO 96625

COMMANDING OFFICER
USS TRIPOLI (LPH-10)
ATTN: METEOROLOGICAL OFFICER
FPO SAN FRANCISCO 96626

COMMANDING OFFICER
USS TARAWA (LHA-1)
FPO SAN FRANCISCO 96622

COMMANDING OFFICER
USS BELLEAU WOOD (LHA-3)
ATTN: METEOROLOGICAL OFFICER
FPO SAN FRANCISCO 96623

COMMANDING OFFICER
USS PELELIU (LHA-5)
FPO SAN FRANCISCO 96624

COMMANDING OFFICER
USS PUGET SOUND (AD-38)
ATTN: METEOROLOGICAL OFFICER
FPO NEW YORK 09544

COMMANDING OFFICER
USS LASALLE (AGF-3)
ATTN: METEOROLOGICAL OFFICER
FPO NEW YORK 09577

COMMANDING OFFICER
USS LEXINGTON (AVT-16)
FPO MIAMI 34088

COMMANDING OFFICER
USS POINT LOMA (AGDS-2)
ATTN: METEOROLOGICAL OFFICER
FPO SAN FRANCISCO 96677

CHIEF OF NAVAL OPERATIONS
OP-952D
U.S. NAVAL OBSERVATORY
WASHINGTON, DC 20390

COMMANDING OFFICER
NORDA
NSTL, MS 39529

COMMANDER
NAVAL OCEANOGRAPHY COMMAND
NSTL, MS 39529

COMMANDING OFFICER (4)
NAVAL OCEANOGRAPHIC OFFICE
BAY ST. LOUIS
NSTL, MS 39522

NAVAL POSTGRADUATE SCHOOL
ATTN: PROF. E. THORNTON
OCEANOGRAPHY DEPT.
MONTEREY, CA 93943

COMMANDER (2)
NAVAIRSYSCOM
ATTN: LIBRARY (AIR-7226)
WASHINGTON, DC 20361

COMMANDER
NAVAIRSYSCOM (AIR-330)
WASHINGTON, DC 20361

COMMANDER, NAVAIRDEVCEN
ATTN: E. BRACKNIS, CODE 203
WARMINSTER, PA 18974

COMMANDER
NAVOCEANSYSCEN
DR. J. RICHTER, CODE 532
SAN DIEGO, CA 92152

COMMANDER
NAVAL SHIP RSCH & DEV. CENTER
SURFACE SHIP DYNAMICS BRANCH
ATTN: S. BALES
BETHESDA, MD 20084

USAFETAC/TS
SCOTT AFB, IL 62225

DIRECTOR (12)
DEFENSE TECH. INFORMATION
CENTER, CAMERON STATION
ALEXANDRIA, VA 22314

END

FILMED